Project Title
Interim Report

TU857
BSc in Computer Science Infrastructure

Student Name: Derry Mahon
Student Number: C22445282

Supervisor: Deirdre Lawless

School of Computer Science
Technological University, Dublin

Date: 01/10/2025

Abstract

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

_____

Student Name

Date

Acknowledgements

I would like to express my sincere gratitude to my project supervisor, Deirdre Lawless, for her expert guidance and invaluable insights throughout this project.
I am also deeply appreciative of all the staff at MGM Partnership for their generous support. Their willingness to complete questionnaires, provide detailed feedback through comments and emails, and participate in on-site testing was instrumental to the practical development and evaluation of the system. Their contributions greatly enriched the quality and relevance of this research.

Contents

1. Introduction

1.1 Project Background

The Architecture, Engineering, and Construction (AEC) sector in Ireland is undergoing a significant digital transformation, driven by technologies that aim to enhance efficiency, safety and compliance standards. As highlighted The Irish Government Build Digital Project Initiative 2030 states *"We believe that through the adoption of digital practices, the Irish construction and built environment sectors will see a transformation in the way we work. Knowledge sharing, information management and transfer will become more streamlined"* [1]. Despite this progress many firms still rely on outdated manual workflows for critical tasks such as project tracking and reporting. These inefficiencies create delays, errors and missed opportunities for proactive decision making and ultimately increase costs and reduce productivity.

My experience working in a Chartered Project Management and Quantity Surveying firm has highlighted these challenges first hand. Manual reporting methods often involve fragmented communication between site and office, particularly in remote or newly developed areas where connectivity is poor. The absence of standardized templates and reliance on manually compiling images, notes, and observations further stretches deadlines and increases the risk of miscommunication and non-compliance this is not only evident in my own experience but is consistently highlighted as a systemic issue across the wider construction industry [2]. The typical workflow for construction project includes numerous stages from data collection to visualization of progress in reports and the feedback loops that support corrective actions and schedule updates (see Figure 1)These operational bottlenecks not only impact project timelines but also inflate costs due to the time intensive nature of reporting.

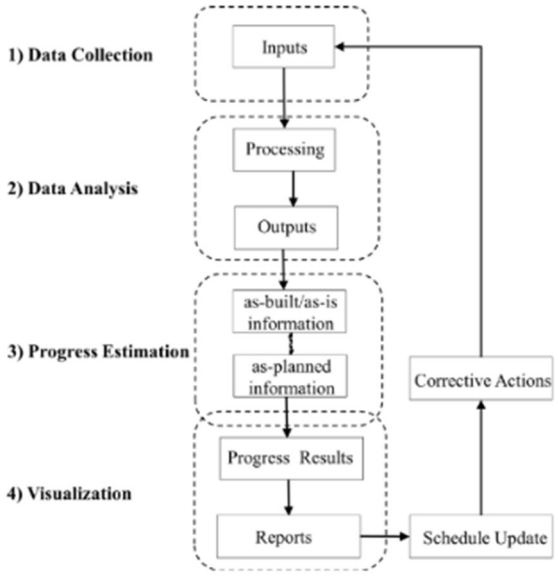

Figure 1 Overview of the current flow of Progress Reporting [3]

The integration of digital tools, particularly AI and mobile technologies, presents a significant opportunity to streamline these workflows.[4] AI and other technologies are increasingly being explored for their potential to automate and enhance construction monitoring and reporting. While we can never remove the human aspect from such a person-

led industry, technology is always evolving to improve accuracy and efficiency in the industry. This is summarised well by Dan Hughes of Alpha Property Insight: *"Technology is really good at number crunching, and then the human will do what the human is good at—ethics, judgement, interpretation, human interaction."*[5]. This reinforces the principle that construction reporting should be technology driven but human led.

The integration of digital tools, particularly AI and mobile technologies, presents a significant opportunity to streamline these workflows. As highlighted in recent literature, the Society of Chartered Surveyors, the leading body of property, land, and construction professionals in Ireland published a report titled *"Will AI help quantity surveying?"* [6]. This feature, written by Mary C. Flynn, Assistant Chief Quantity Surveyor at Dublin City Council, explores the use of AI tools and how they are reshaping the construction industry. Kathy Bloomgarden, CEO of Ruder Finn, is further quoted in this article endorsing the idea of optimising workflows in the industry: "*We are in a moment of unprecedented human potential, and AI is the key to unlocking that potential*." [6]. These perspectives highlight the growing consensus that digital transformation is essential for future proofing the AEC sector.

I am currently working in the construction sector, where my responsibilities include leveraging technology to improve workflows and efficiency in a sector which lags behind others in technological adaption [7]. I focus on exploring and implementing digital solutions, particularly AI driven tools, to enhance reporting and project management processes. In addition to this technology-focused work, I also assist with site visits, which gives me practical insight into the challenges faced on site, such as fragmented communication and delays caused by manual reporting. This combination of technical expertise and on-site experience has motivated me to develop ReportBuilderPro an integrated web and mobile application that combines mobile accessibility, AI driven insights, and real-time data synchronisation. By addressing these challenges, the platform aims to deliver accurate, user-friendly reporting while remaining scalable and adaptable across diverse AEC contexts. Incorporating user feedback and iterative testing will ensure the solution is grounded in practical needs and capable of driving meaningful digital transformation in the industry. Despite the availability of digital tools, existing solutions often fail to integrate offline-first reporting, AI-driven analysis, and standardised templates within a single tool. This creates a gap for a unified, intelligent reporting application designed for real-world construction site constraints.

1.2 Project Description

ReportBuilderPro will be an integrated digital reporting system developed specifically for the construction industry. It will consist of both a web and mobile application, designed to streamline progress reporting, health and safety documentation and risk identification through real time data capture and AI enhanced analysis. The system will address key challenges in current workflows, such as manual data entry, inconsistent reporting formats, and poor connectivity on remote sites.

The Web Application aspect will serve as the central control hub for report template creation, refinement, project oversight and client deliverables. It will feature a dashboard which will display data taken from our report scans that allows users to monitor project risks, timelines and possible shortages of materials. Reports which are submitted from the field will be refined and can be ran through our AI scanner which will be able to classify and highlight

potential issues in a visual dashboard, providing insights and supporting proactive decision making before generating a professional PDF output for client delivery.

The Mobile Application will be tailored for on-site usage by project managers, quantity surveyor or other expert personnel. It will support offline-first data capture, enabling users to document progress, upload images, dictate notes and complete safety checklists even in areas with limited connectivity. Once online, the app will sync with the backend to update the report for further completion on the web application.

A key innovation in ReportBuilderPro will be its integration of AI technologies to enhance reporting intelligence. The system will use Natural Language Processing (NLP) to automatically scan report text for risks, delays, and material shortages. Computer vision models will assess uploaded images to detect safety hazards and progress indicators. These AI tools will flag issues, identify patterns across projects, and provide smart recommendations to users, such as highlighting missing information or suggesting corrective actions. The AI models will improve over time through continuous learning, ensuring that insights become more accurate and context-aware with each use. Importantly, the system will be technology driven but human-led, supporting professionals in making faster, more informed decisions while maintaining control over interpretation and judgment.

1.3 Project Aims and Objectives

The overall aim of the project is to design and develop an integrated digital reporting system comprising a web application and a mobile application that enhances the efficiency, accuracy, and accessibility of progress reporting, health and safety documentation, and material shortage and risk identification in the construction industry. The system will leverage offline-first mobile data capture and AI driven analysis to streamline reporting workflows and support timely, data informed decision-making.

The scope of this project will focus on reporting workflows within construction site operations, specifically progress tracking, health and safety documentation, and material shortage identification. It will not extend to full project management or financial planning modules.

Project Objectives

- **Conduct a comprehensive review of existing technologies** to identify current capabilities, limitations and opportunities for innovation particularly in the AI integration and offline functionality.
- **Conduct Expert Elicitation** through questionnaire as well as direct observation of real-world workflows to ensure the system addresses practical, on site and in office needs
- **Appropriate System Architecture** that separates the web application, mobile application, backend services and Ai component, ensuring scalability maintainability and ease of integration
- **Develop a web application** that enables users to create and refine customisable report templates, visualise flagged risks, delays and material shortages via dashboard

- **Develop a mobile application** that allows onsite users to access and complete report templates, capture images dictate notes and operate offline with automatic synchronisation upon reconnection
- **Implement a robust backend** using appropriate technologies to manage data storage user authentication and file handling efficiently and securely
- **Integrate a Natural Language Processing** component to analyse report text, classify content into relevant categories such as delays, risks, and material shortages, and flag potential issues within the system's dashboard adopting the technologies most appropriate for the project.
- **Evaluate the system** through functional, usability, and performance testing using real user feedback to assess accuracy, ease of use, offline performance, and the quality of generated reports. Additionally, evaluate the usefulness of the system using feedback from at least three domain experts to ensure practical relevance and industry alignment.

1.4 Project Scope

This project focuses on the design and development of a digital reporting system for the construction industry, comprising of a web application and a mobile application. The scope includes core functionalities such as template creation, offline first data capture, Ai driven report analysis and PDF generation. The system is intended for use by Quantity surveyors and Project managers and other site personnel to streamline progress reports, health and safety documents and risk identification. Key constraints include intermittent internet connectivity on construction sites, varying levels of digital literacy among users, and the need for AI outputs to serve as advisory rather than authoritative recommendations.

What is in the Project Scope?

- Development of a web application for report template creation, dashboard monitoring, and PDF exporting.
- Development of a mobile application for on-site data entry, including image uploads and voice notes.
- Implementation of offline-first functionality for mobile users.
- Integration of a Natural Language Processing (NLP) engine to analyse report text and flag risks, delays, and material shortages.
- Setup of a backend infrastructure for data storage, user authentication, and file handling.
- Functional, usability, and performance testing using real user feedback from a construction company environment.
- Delivery of system documentation, architecture diagrams, and a final evaluation report.

What is out of Project Scope

- Full-scale deployment across multiple companies or commercial environments.
- Integration with third-party enterprise systems (e.g., ERP, BIM platforms).
- Real-time collaboration features (e.g., multi-user editing or live chat).
- Support for regulatory compliance across different countries or regions.
- Mobile app publication to public app stores (e.g., Google Play, Apple App Store).

- Long-term maintenance, updates, or scalability planning beyond the prototype phase.

1.5 Thesis Roadmap

1. **Introduction**: Discusses the projects background, motivation, aims, and objectives, and outlines the scope of the work, providing a roadmap for the thesis.
2. **Literature Review**: This chapter explores existing solutions and research relevant to construction reporting tools and similar applications. It will also it will examine technologies, methodologies for the technical aspects of this project along with their strengths and limitations.
3. **System Design**: Provides an in-depth look at the architectural and design choices, explaining the selected software methodology, system components, and how each part of the application will interact to deliver a cohesive user experience.
4. **Testing and Evaluation**: Details the comprehensive testing strategy, including unit, integration, and user testing, and describes the methods used to evaluate the project's performance, usability, and reliability.
5. **Prototype Development**: Documents the development of the initial prototype and the current state of the project, this chapter will also include code snippets and screenshot of the current working version.
6. **Issues and Future Work**: Reviews the project's current limitations, potential risks, and outlines areas for future development that need to be completed for achieve the end goal of this project.

2. Literature Review

2.1 Introduction

This section reviews existing literature, technologies, and systems relevant to the development of a digital reporting solution for the construction industry based upon my research conducted in Section 1. It explores current software tools used in the field, evaluates the technologies selected for implementation, and examines domain-specific research on construction reporting and AI integration. Additionally, it considers similar final-year projects to position this work within the academic context and identify areas for innovation.

While the original idea stemmed from my personal experience working within the construction sector, it became clear that the challenges I observed such as inefficiencies, fragmented workflows, and lack of user-friendly digital tools were not isolated. These issues are echoed across the industry and are well-documented in EY's *Detailed Description of Needs for the Irish Construction/Built Environment Sector* [8] under Project Ireland 2040. Ireland's long-term, overarching strategy for social, economic, and cultural development, combining the National Planning Framework (NPF) [9] and the National Development Plan (NDP)[10] to guide the country's future to 2040

The report highlights a stark contrast between firms that are embracing innovation and those that are not. For example, *"The rapid pace of technological change is already seeing some early adopters in the construction industry adopt new technological processes and more advanced construction systems to deliver value to their clients."* [8]. It also notes *"40% of firms in the industry are not using any form of automated technologies within their current projects."* [8]. This divide reinforces the need for a user-friendly system one that lowers the barrier to entry for small and medium-sized firms, supports digital adoption, and enables collaboration across the supply chain.

2.2 Alternative Existing Solutions

As part of my research, I explored the core technologies currently driving the Irish construction industry. These tools are actively employed by some of the country's leading firms to deliver multi-million-euro developments and play a vital role in supporting every phase of the project lifecycle.

Tool 1: Procore

Procore is a comprehensive construction management platform offering tools for project documentation, budgeting, scheduling, and collaboration. It is one of the most widely adopted tools across the AEC industry.

URL: https://www.procore.com

- *Strengths*: Procore offers an integrated suite of project management tools, strong mobile support and seamless cloud accessibility, enabling efficient coordination across multiple stakeholders.

- *Weaknesses*: However, the platform is relatively expensive has limited offline capabilities and presents a complex user interface that may not be well-suited to smaller teams or firms with limited technical experience.
- *Relevance*: ReportBuilderPro will improve accessibility and simplifies reporting for smaller teams.

Tool 2: SiteCam

A mobile first site documentation tool designed for capturing progress through photographs and notes.
URL: https://sitecam.io/

- *Strengths*: SiteCam excels in image-based reporting and offers a user-friendly mobile interface, making it ideal for quick on-site documentation.
- *Weaknesses*: The platform lacks advanced template customization and does not support AI-driven analysis, which limits its scalability for more complex reporting needs.
- *Relevance*: ReportBuilderPro will significantly expand SiteCam's functionality by introducing AI powered risk detection and intelligent reporting workflows. Its flexible templates and structured data capture will make it ideal for teams needing more than just visual documentation especially where compliance and safety insights are critical.

Tool 3: PowerProject

Power Project is used for scheduling, managing of resources, and track the progress of construction projects widely used for planning purposes.
URL: https://www.elecosoft.com/products/powerproject

- *Strengths:* Offers advanced Gantt chart scheduling, robust resource management, and BIM integration. It is well-established in the industry for planning and tracking project timelines.
- *Weaknesses:* Not designed for health and safety reporting or on-site data capture. Lacks mobile-first functionality and does not support AI driven analysis or automated reporting workflows.
- *Relevance*: While PowerProject excels in planning and scheduling, it lacks capabilities for field-level reporting and safety oversight. ReportBuilderPro will fill this gap with real time, AI enhanced reporting, offline usability, and intelligent risk flagging bringing actionable insights directly from the site to decision makers.

Tool 4: Fieldwire

Fieldwire is a site management solution developed for the field, enabling you to share the right information with your teams and coordinate on-site in real time.
URL: https://www.fieldwire.com

- *Strengths*: Offers robust task tracking and drawing management features, enabling efficient communication and collaboration among field teams.
- *Weaknesses*: Lacks dedicated functionality for health and safety reporting, automated reporting workflows, and customizable templates, which can limit its effectiveness for compliance-driven projects.

- *Relevance*: ReportBuilderPro will fill this gap by delivering structured, AI powered reporting and automated safety compliance tools. Its intelligent templates and real time insights provide smaller teams with the ability to generate high quality reports effortlessly, enhancing both accuracy and decision-making on-site.

Comparative analysis of my System vs Industry Systems

Feature	ReportBuilderPro	Procore	SiteCam	Powerproject	Fieldwire
Offline-first data capture	Yes	No	Yes	No	Yes
AI issue finder	Yes	No	No	No	No
PDF report generation	Yes	Yes	Yes	No	Yes
Template customization	Yes	Yes	No	No	Yes
Real-time dashboard	Yes	Yes	No	Yes	Yes

2.3 Technologies Researched

This section outlines the technologies selected for the development of my reporting system. Each component was chosen based on its suitability for rapid prototyping, scalability and compatibility with the systems' functional requirements, particularly offline first mobile reporting and AI enhanced analysis.

2.3.1 Frontend Technologies

"In the dynamic realm of web development, the front-end serves as the gateway to user interaction, making the selection of appropriate technologies a pivotal decision for developers and organizations." [11]

To support the goals of Report Builder Pro, the frontend must deliver a responsive, modular interface for building and managing reports, with smooth user interactions and real-time updates. These are the front-end technologies I researched.

React.js: *(Selected) URL https://react.dev/*

A widely adopted JavaScript library for building dynamic user interfaces. React's component-based architecture supports modular design and efficient rendering, making it ideal for building the dashboard and report template builder in the web application.

Vue.js:

A progressive Javascript framework known for its simplicity and ease of integration. Vue is lightweight and used mainly in smaller projects

Angular:

A full-featured frontend framework developed by Google, offering built-in routing, state management, and testing tools.

For the frontend of Report Builder Pro, I will select React.js due to its component-based architecture, which supports the creation of dynamic, reusable UI elements ideal for building an interactive report template builder and dashboard. These technologies were chosen for their strong documentation, active communities, and ability to support rapid development within the limited timeframe of a dissertation project.

2.3.2 Backend Technologies

The backend must support secure user authentication, scalable processing of diverse data types including text, images, and structured inputs and efficient communication with the NLP engine. As technology continues to advance, backend innovation will be pivotal in supporting sustainable, secure, and intelligent digital infrastructure.[12] Particularly for ReportBuilderPro the importance of reliability and security is of the upmost importance.

Node.js: *(Selected) URL https://nodejs.org/en*

A lightweight and scalable runtime environment for building server-side applications. It will be used to manage user authentication, report submission, and communication between the frontend and backend.

FastAPI: *(Selected) URL https://fastapi.tiangolo.co*

A modern Python framework for building APIs, chosen specifically for integrating the NLP engine. FastAPI supports asynchronous operations and is optimized for performance.

Django:

A high-level Python web framework with built-in admin tools and ORM support. It's well-suited for traditional web applications.

Flask:

A lightweight Python micro-framework used for building simple web applications and APIs.

For the frontend of Report Builder Pro, I will use Node.js to handle user authentication and manage communication between the frontend and backend efficiently, while FastAPI is selected for its high performance and compatibility with Python-based NLP tools, making it well suited for integrating the natural language processing engine. These technologies were chosen for their strong documentation, active communities, and ability to support rapid development within the limited timeframe of a dissertation project.

2.3.3 Database

ReportBuilderPro will require a database solution that can handle a wide variety of data types, including structured user information, unstructured report content, and AI generated analysis results. Flexibility in schema design and scalability are key, given the diverse nature of the data being processed during report generation.

MongoDB: *(Selected) URL https://www.mongodb.com/*

A NoSQL document-oriented database that supports flexible schema design. It is well-suited for storing varied report formats, user data, and AI analysis results.

PostgreSQL:

A powerful relational database known for its reliability and support for complex queries

Firebase:

A cloud-hosted NoSQL database optimized for mobile apps with real-time syncing.

For ReportBuilderPro, MongoDB will be selected as the primary database due to its document-oriented structure, which allows for flexible storage of varied report formats and NLP outputs. Its ease of integration with Node.js and support for rapid development made it the most suitable choice for ReportBuilderPro.[13]

2.3.4 Ai and NLP Tools

To enable intelligent analysis of construction reports, Report Builder Pro will incorporate natural language processing (NLP) and machine learning tools that will extract insights from unstructured text. These tools will be used to identify risks, delays, and material shortages based on patterns in user-generated content.

spaCy: *(Selected) URL https://spacy.io/*

A robust Python library for natural language processing, used for text preprocessing such as tokenization, entity recognition, and sentence parsing.

Scikit-learn: *(Selected) URL https://scikit-learn.org/stable/*

A machine learning library used to train and evaluate classification models that flag risks, delays, and material shortages in report text.

NLTK:

A classic Python library for natural language processing, widely used in academic settings.

PyTorch:

Deep learning frameworks used for building complex AI models.

spaCy will be selected for its efficient text preprocessing capabilities, including tokenization and entity recognition, while Scikit-learn will be used to train lightweight classification models that flag key issues in report text. These libraries will support rapid experimentation and integration with the Python-based backend.

2.3.4 PDF Generation

To produce professional, client-facing reports, Report Builder Pro will require a solution for converting structured HTML templates and AI enhanced content into downloadable PDF documents.

pdfkit: *(Selected) URL https://pdfkit.org/*

A Python library used to convert HTML content into PDF format. It will be used to generate professional client-facing reports from the refined templates and AI analysis results.

pdfkit will be used to handle this conversion, offering a simple and effective way to generate consistent, well-formatted PDFs directly from the backend.

2.3.5 Cloud Hosting Services

Cloud services will be used to host the application and store user-generated content, including images and reports. While long-term scalability is not a requirement, reliable deployment and secure storage will be essential during development and testing.

AWS (Amazon Web Services): *(Selected) URL* https://aws.amazon.com/

Used for secure file storage, particularly for images and report documents uploaded via the mobile app.

Azure: *(Selected) URL https://azure.microsoft.com/en-us/*

Selected for hosting the web application and backend services due to its integration with development tools and scalability options.

Azure will be selected to host the web application and backend services due to its integration with development tools and ease of deployment. AWS will be used for storing uploaded files, particularly images and PDFs, thanks to its robust and secure storage options.

2.4 Other Research

2.4.1 Expert Elicitation

Conducting an elicitation was a key part in finalising my system requirements. It allowed me to gather informed insights from both field experts and real users ensuring the system was grounded in the practical needs of the user. One particularly useful framework to this was *A practical guide to structured expert elicitation using the IDEA Protocol* by Victoria Hemming et al[14].

The IDEA Protocol (Investigate–Discuss–Estimate–Aggregate) is well regarded for its ability to reduce bias and improve the reliability of expert input.

Before applying the protocol, I carried out essential pre-elicitation steps, which included:

- Compiling relevant background information to inform participants
- Identifying and contacting suitable experts across construction and technology domains
- Briefing participants on the elicitation process, its purpose, and how their input would shape the system design

These steps ensured that the elicitation was well-informed and that participants were prepared to contribute meaningfully. The structured nature of the IDEA Protocol helped me refine system features such as reporting workflows, risk detection mechanisms, and template flexibility, aligning them with real-world expectations and industry standards. As stated by Hemming in the piece, *"The protocol is designed to reduce overconfidence, encourage diversity of opinion, and improve the accuracy of group judgements."* [14]. Highlighting the benefit to using IDEA over informal consultation to ensuring my finalised system requirements met user needs.

Overview of the IDEA Protocol [14]

The IDEA protocol stands for:

- **Investigate**: All experts individually answer questions and provide reasoning on their judgement.
- **Discuss**: Experts share reasoning and insights to understand differing views.
- **Estimate**: Experts revise their judgments based on the discussion.
- **Aggregate (Post-elicitation)**: Final judgments are combined to form a collective view.

This protocol is designed to reduce bias, improve the quality of expert input, and ensure transparency in decision-making. It's particularly useful when designing systems that rely on domain-specific knowledge, like ReportBuilderPro does.

How I applied IDEA Protocol

Investigate

- I began by identifying key stakeholders in my userbase such as Project Managers and Quantity Surveyors.
- Each expert was asked to complete a questionnaire individually, focusing on their current reporting practises, pain points, and desired features.

Discuss

- After I collected the initial responses, I followed up with informal interviews and emails to further clarify the data I had collected
- This stage helped me gain deeper insights to the workflows challenges and prioritise features

Estimate

- Based on the discussions, I asked the experts to revisit initial feedback and refine their input

- This iterative feedback helped me shape a more accurate and prioritised set of requirements

Aggregate

- I complied my results into a consolidated requirements list, identifying common themes and critical needs.
- These aggregated insights directly informed me of the system specifications and design decisions

How it strengthened my Project

- The IDEA Protocol added rigor to my requirements gathering process.
- It ensured the system was grounded in the real experts needs and not based off assumption
- It provided me with a defensible methodology for stakeholder's engagement which strengthened the credibility of this project

2.4.2 Feature Driven Development (FDD)

In order to guide the development of my system, I adopted a Feature-Driven Development approach. FDD is a client centric, iterative approach that focuses on designing and building software around clearly defined user valued features. This approach aligns well with my project goals.

'Feature Driven Development (FDD) and Agile Modeling'. An article by Scott W. Ambler an international keynotes speaker [15]. Outlines the methodology as built on five key activities:

1. Develop an overall model
2. Build a feature list
3. Plan by feature
4. Design by Feature
5. Build by feature

By implementing this structure, it ensures the system is built in both systematic and responsive to the user's needs.

Overview of FDD Principles[15]

- Feature-centric: Each feature represents a small, client-recognizable piece of functionality
- Iterative and incremental: Features are developed in short cycles, allowing for continuous refinement.
- Domain-driven: The system is modelled around real-world entities and processes, such as reports, users, and safety flags.
- Team-oriented: Roles such as Feature Owner and Class Owner help ensure accountability and collaboration.

As stated in the reference from Ambler, *"FDD blends several industry-recognized best practices into a cohesive whole. These practices are driven from a client-valued functionality (feature) perspective."* [15]

Applying FDD to my Project

Develop an Overall Model

- Map out core entities in the system i.e. users, reports, templates, risk flags and workflows.
- This model was informed by expert input gathered through the IDEA Protocol in Section 2.4.1.

Build a Feature List

- Using the elicitation results, I compiled a list of desired features.

Plan by Feature

- Features are prioritised based on user feedback and feasibility.
- Group features into logical modules

Design by Feature

- Each feature is designed with user experience in mind, focusing on simplicity, accessibility, and relevance to field workflows.

Build by Feature

- The system architecture and design decisions are structured to support modular, feature-based development.

2.5 Existing Final Year Projects

As part of this research, I accessed the university library to review previous projects, focusing on work that demonstrated advanced mobile and web development, innovative applications of machine learning, progress tracking or reporting systems and projects that incorporated complex or multi database architecture.

*Note all past projects were accessed from the TU Dublin Library Catalogue [16]

2.5.1 Project One

Title: Automatic License Plate Detection using Image Processing

Student: Ayan Abedin

Description (brief): The project aims to develop a robust and efficient system for real-time Automatic License Plate Recognition (ALPR) using OpenCV, Python programming, and advanced computer vision techniques. The primary objective is to create a comprehensive solution that automates the identification and processing of license plates within parking management systems. The system uses a computer/Raspberry Pi and a camera for image capture, employing OpenCV and Python for computer vision algorithms. It follows stages of image capture, quality enhancement, identifying license plate areas, character segmentation, and recognition, prioritizing real-time processing on the Raspberry Pi.

What is complex in this project:

The creation of a real time image recognition system which I could implement for risk delays and material shortage based off trends and my report and images.

What are the key aspects and complexity in this project:

What's described in the complexity is a key strength and selling point of this project however this is already a widely used and accessible technology already in use by An Garda Siochana for example.

2.5.2 Project Two

Title: Anseo [17]

Student: Johnathan Hew

Description (brief):

The goal of this project is to create an efficient, user-friendly solution that not only addresses the limitations of the pen and paper method but also offers additional features, such as reporting and data visualization. These added functionalities enable educators to better understand attendance trends and make informed decisions to improve student engagement and performance. To achieve this, the Anseo application will be developed using PostgreSQL, Express, React and Node using Feature Driven Development (FDD), an agile framework that focuses on delivering features incrementally and adapting to feedback throughout the development process. By implementing Anseo, institutions can streamline the attendance management process and leverage data-driven insights to enhance the overall educational experience.

What is complex in this project:

- Data Visualisation and Reporting
- Real Time Data Handling
- Scalability and Performance
- Feature Driven Design

What technical architecture was used:

- Postgres SQL Relational database for storing attendance records and other user data
- Express.js backend framework for API requests and server-side logic
- React Frontend Library for dynamic user responsive interface
- Node.js Runtime environment for executing JavaScript on server

What are the key strengths and weaknesses associated with this project:

User centric design, Data Drive insights and Modern tech stacks are clear strengths in this project and will be implemented in mine too.

Some weaknesses may be the complexity in Visualisation Aswell as the use of FDD which could misalign the priorities of the features if not careful

2.5.3 Project Third

Title: Deep [18]

Student: Bryan McCarthy

Description (brief):

The “Deep” project is a web application designed to enhance productivity and time management by minimizing distractions and promoting focused work sessions. It targets university students and incorporates tools like task management, calendar scheduling, Pomodoro timers, and data analytics to help users track and improve their study habits. The application is built using a modern tech stack including TypeScript, React.js, Vite, SASS, Go, Gin, GORM, and PostgreSQL. It follows Agile development with a Scrum framework to ensure iterative progress and adaptability to feedback.

What is complex in this project:

- Real-Time Task Tracking and Estimation: Estimating time remaining based on difficulty, progress, and time spent.
- Data Visualization: Dashboard analytics showing time allocation and milestones.
- User Experience Design: Minimalist, distraction-free interface with responsive design.
- Agile Development with Scrum: Managing iterative development and backlog prioritization.
- Session Management and Secure Routing: Using Gorilla sessions and Gin framework for secure backend operations.

What technical architecture was used:

- Frontend: React.js and Vite a responsive and component-based UI.
- Backend: Go with Gin framework for routing and GORM for ORM.
- Database: PostgreSQL for storing user data and task records.
- Session Management: Gorilla/sessions for secure cookie-based sessions.
- Local Storage: Used for storing less critical data like timer settings.

What are the key strengths and weaknesses associated with this project:

Strengths:

- User-Centric Design: Clean, minimalist interface focused on reducing distractions.
- Comprehensive Productivity Tools: Task management, calendar, Pomodoro timer, and dashboard analytics.
- Modern Tech Stack: Efficient and scalable technologies with strong community support.
- Agile Methodology: Scrum framework allows for iterative development and responsiveness to feedback.

- Strong Conceptual Foundation: Inspired by Cal Newport's "Deep Work" and GTD methodology, aligning features with cognitive science.

Weaknesses:

- Complexity in Estimating Time Remaining: The algorithm for predicting task completion time may need refinement.
- Potential Overhead in Tech Stack: Using Go and Gin may be overkill for simpler applications unless scalability is a priority.
- FDD Not Used: Unlike Anseo, this project uses Scrum, which may be more flexible, but less feature focused.

2.6 Conclusions

Based on my research ReportBuilderPro should be designed as scalable, intelligent reporting solution for construction and project environments, built on a carefully selected tech stack that will balance performance, usability and extensibility. Using spaCy for preprocessing and Scikit-learn for model development ensures robust natural language processing. Fast API acts as the API layer that wraps the NLP to the rest of the system serving as the bridge between frontend and backend while PostgreSQL offers a structured, relational database environment with strong data integrity and advanced querying capabilities—especially well-suited for analytics and reporting workflows. React powers the dashboard for dynamic, user-friendly interaction with the data. Hosted on Azure, the platform benefits from enterprise-grade reliability and scalability. Development follows Feature Driven Development (FDD), allowing incremental delivery of high-value features. Requirements were elicited through analysis of existing reporting workflows, user interviews, and benchmarking against tools like SiteCam, Procore, Fieldwire and ProjectPlanner ensuring the solution is grounded in real-world needs and optimized for construction reporting challenges.

3. System Analysis

3.1 System Overview

ReportBuilderPro is a web-based application designed to streamline construction site reporting. It enables site managers and engineers to quickly create, manage, and export structured reports using natural language input. With a user-friendly dashboard, real-time data visualization, and automated PDF generation, the system enhances productivity and ensures consistent, high-quality documentation.

3.2 Stakeholder and User Analysis

3.2.1 Stakeholder Identification

Stakeholder identification is critical to the functional success of software development projects [19] to ensure the systems design reflects the real-world needs and expectations. This is particularly relevant for ReportBuilderPro, given its user-centric design approach, which prioritises the requirements and workflows of construction professionals to deliver a practical and intuitive solution.

During those initial stages I identified the following stakeholder groups

Primary Stakeholders

- Project Managers
- Quantity Surveyors
- Company Directors
- IT Admin

Secondary Stakeholders

- Clients

To ensure my system met the needs of real-world experts I carried out expert elicitation using the IDEA Protocol [14] as outlined in Section 2.4.1 of this report.

In addition to a formal elicitation, I also drew on

- First hand experience with current workflows.
- Competitor analysis to benchmark features and identify gaps

3.2.2 Stakeholder Needs

Each stakeholder group has distinct priorities and expectations. Based on my expert elicitation and follow up conversations the main needs of stakeholders were identified as follows

- **Project Managers** Require real-time progress reporting, risk identification, and streamlined health and safety documentation.
- **Quantity Surveyors** Need accurate material tracking and cost-related reporting integrated into site workflows.
- **Company Directors** Seek high-level summaries and compliance assurance for decision-making.

- **IT Administrators** Focus on system security, data integrity, and ease of deployment.

3.2.3 Workflow Pain Points

Current reporting workflows in construction face systemic challenges that hinder efficiency and compliance. This section highlights these pain points, forming the basis for ReportBuilderPro's design objectives.

- Manual compilation of images, notes, and observations leading to delays and miscommunication.
- Lack of standardised templates across projects, increasing risk of non-compliance.
- Intermittent connectivity on construction sites, making real-time reporting difficult.
- Limited integration of AI-driven insights in existing tools, resulting in missed opportunities for proactive risk management.

3.3 Requirements Gathering and Elicitation

This section outlines the functional and non-functional requirements for the proposed site reporting system. The analysis is grounded in best practices from software engineering and user-centred design and considers emerging technologies such as natural language processing (NLP) to support intelligent reporting workflows in construction management.

3.3.1 Methods Used

The requirements gathering process for ReportBuilderPro was designed to be rigorous and stakeholder driven. To achieve this, I adopted the IDEA Protocol [14], a structured elicitation method that reduces bias and improves the accuracy of expert input.

Overview of the IDEA Protocol [14]

- **Investigate** All experts individually answer questions and provide reasoning on their judgment.
- **Discuss** Experts share reasoning and insights to understand differing views.
- **Estimate** Experts revise their judgments based on the discussion.
- **Aggregate** Final judgments are combined to form a collective view.

This protocol is particularly useful for systems like ReportBuilderPro that rely on domain specific knowledge, as it ensures transparency and diversity of opinion.

How I Applied the IDEA Protocol

- **Investigate** I began by identifying key stakeholders such as Project Managers and Quantity Surveyors. Each expert completed an individual questionnaire focusing on current reporting practices, pain points, and desired features.
- **Discuss** After collecting initial responses, I conducted informal interviews and email follow ups to clarify and deepen insights into workflow challenges.
- **Estimate** Based on discussions, experts revisited their initial feedback and refined their input, helping shape a more accurate and prioritised set of requirements.
- **Aggregate** I compiled results into a consolidated requirements list, identifying common themes and critical needs. These aggregated insights directly informed system specifications and design decisions.

How It Strengthened My Project

- **Added rigor** The IDEA Protocol provided a structured, defensible methodology for requirements gathering.
- **Grounded in real needs** Ensured the system was based on expert input rather than assumptions.
- **Improved credibility** Demonstrated a transparent and systematic approach to stakeholder engagement.
- **Clarity of Features** The data shows a clear trend as to where the workflow can be improved and provided insight on what ReportBuilderPro's functionality should be.

3.3.2 Key Findings

The elicitation process highlighted several critical requirements:

- **Standardised templates** to reduce inconsistencies and compliance risks.
- **Offline first functionality** Essential for sites with intermittent connectivity.
- **AI driven analysis** to classify delays, risks, and material shortages automatically.
- **User friendly interface** Designed for varying levels of digital literacy among site personnel.
- **Secure data handling** to protect sensitive project information during sync and storage.

3.4 Functional and Non-Functional Requirements

3.4.1 Functional Requirements

Structured Report Generation via Web Interface

- The system shall enable users to create site reports using a structured, simple interface.
- Justification: Structured input ensures consistency and facilitates data validation,

Natural Language Processing (NLP) for Report Analysis

- The system shall incorporate NLP techniques to analyse free text entries.
- Justification: Reduces cognitive load on users and enhances report readability; aligns with current trends in intelligent document processing.

PDF Export Capability

- Users shall be able to export reports in PDF format for offline access, sharing, and archival.
- Justification: PDF is a widely accepted format in professional and regulatory contexts.

Interactive Dashboard for Report Management

- The system shall provide a dashboard for viewing, filtering, and analysing reports.

- Justification: Enhances situational awareness and supports decision-making through visual analytics.

User Authentication and Role Based Access Control

- The system shall implement secure login and access control mechanisms based on user roles.
- Justification: Ensures data confidentiality and integrity; aligns with principles of least privilege.

3.4.2 Non-Functional Requirements

Performance

- NLP processing and dashboard interactions shall exhibit low latency
- Justification: High responsiveness is critical for user satisfaction and real-time decision-making.

Security

- Data shall be transmitted over HTTPS and stored in encrypted databases
- Justification: Protects sensitive information and ensures compliance with data protection regulations (e.g., GDPR).

Mobile Responsiveness

- The system shall be optimized for mobile devices to support on-site usage.
- Justification: Field operatives require access to reporting tools in dynamic, mobile environments.

3.4.3 Requirement Prioritisation (MoSCow)

The requirements matrix provides a clear link between functional requirements and user needs, helping prioritise development tasks. Each requirement was derived directly from the use case specifications, ensuring that the system design reflects real-world workflows and user goals.

Req ID	Name of Req	Description	Priority	User Contact
R!	Report Creation	Create and edit site reports	High	PM
R2	Image Tagging	AI-based photo content detection	Medium	PM
R3	Offline Sync	Save and sync data offline	High	PM
R4	PDF Export	Generate client-ready reports	High	QS and PM
R5	Dashboard View	Real-time project tracking	Medium	Admin
R6	NLP Risk Detection	Flag risks, delays, shortages	High	Admin

Figure 2 Requirements Specification Matrix

3.5 Use Case Analysis

Use cases are critical in software development because they provide a structured way to capture functional requirements from the user's perspective. They ensure that the system design reflects real-world workflows rather than abstract technical features. *"The impact of use case diagrams on software development cannot be overstated. Their ability to simplify complex requirements, clarify system boundaries, and ensure user-centered design contributes directly to the success of software projects."*[20] . Use cases improve requirements elicitation, stakeholder communication, and system design clarity, making them essential to building user centred applications.

For ReportBuilderPro, use cases are particularly important because the system operates in a dynamic, high-pressure environment the construction site. By modelling tasks such as creating templates, documenting on site progress, submitting reports for AI analysis, and reviewing insights, we ensure that the application supports actual user goals while remaining intuitive for users who may not be highly tech-literate. This approach reflects ISO standards such as suitability for the task and error tolerance[21]. It also aligns with Shneiderman's Golden Rules by emphasizing consistency, informative feedback, and easy reversal of actions.[22] Together, these standards help reduce complexity, minimize training requirements, and promote adoption across diverse teams.

3.5.1 System Level Use Case

3.5.2 User Creates Report Template

This use case ensures consistency and compliance by allowing managers to design standardised templates tailored to project and regulatory requirements.

USE CASE	1	Project Manager Creates Standardised Report Template
Description of Goal in Context	Enable the Project Manager to create a standardized report template for site documentation, ensuring consistency and compliance across projects.	
Preconditions	Manager is authenticated in the web application. Manager has appropriate permissions to create templates. Web application is operational.	
Post Conditions, Success End Condition	Template is saved in the system and available for assignment to projects. Confirmation message displayed to the manager.	
DESCRIPTION of Scenario	The Project Manager logs into the web application, navigates to the template builder, and creates a new report template by adding sections (e.g., progress photos, safety checks, notes). Once finalized, the template is saved and becomes available for use by on-site personnel..	
Main Flow		
Step	**Action**	**Response**
1	Log into web application	Authentication verified
2	Navigate to Template Builder	Display template creation interface
3	Add sections (images, notes, checklists)	System validates input
4	Finalise template configure any mandatory fields	System applies settings
5	Save template	Confirmation message displayed
6	Template available for use	System updates template list
EXCEPTIONS or ERROR Flow Description		
Step	**Branching Action**	**Alternate**
1	App crashes during template creation	Auto-save draft and prompt recovery.
ALTERNATIVE or VARIATION Flow Description Variations in how the report is submitted depending on connectivity		
Step	**Branching Action**	**Alternate**
1	Manager edits an existing template instead of creating a new one.	
2	Manager duplicates an existing template for faster setup.	

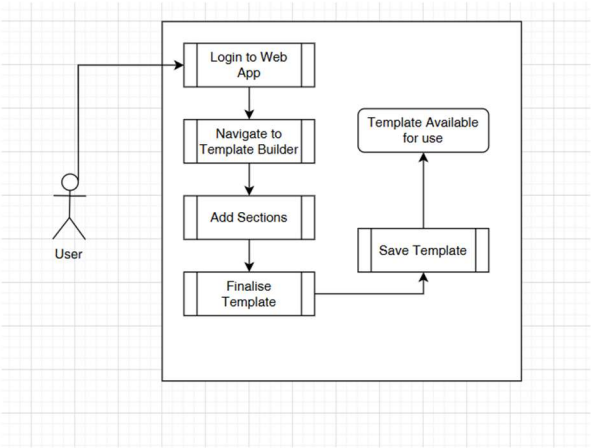

Figure 3 User Creates Report Template

3.5.3 User Fills in Report On-Site

Capturing real time site data through structured templates improves accuracy, reduces reporting delays, and provides a solid foundation to create the finalised report.

USE CASE	2	Project Manager uses predefined report template to document site progress
Description of Goal in Context	Enable the Project Manager to document site progress during a visit by completing a predefined report template, capturing images, and adding observations for later AI analysis and finalisation for client delivery.	
Preconditions	Mobile app installed and authenticated. Assigned report template available. Device has sufficient storage for offline mode.	
Post Conditions, Success End Condition	Report saved locally or synced to cloud. Confirmation message displayed to user.	
DESCRIPTION of Scenario	The Project Manager opens the mobile app on-site, selects the assigned report template, captures progress photos, and adds notes or dictated observations. If offline, the report is stored locally and queued for sync. Once online, the report is uploaded to the cloud for finalisation on web application.	

Main Flow

Step	**Action**	**Response**
1	Open mobile app	Verify authentication
2	Select assigned report template	Display template fields
3	Capture site images	Store locally
4	Add notes or dictate observations	Validate input
5	Save report	Confirm save
6	Sync report to cloud	Show sync progress & success

EXCEPTIONS or ERROR Flow
Description

Step	**Branching Action**	**Alternate**
1	App crashes during report creation	Auto-save draft and prompt recovery.
	Image upload fails	Retry prompt.

ALTERNATIVE or VARIATION Flow
Description
Variations in how the report is submitted depending on connectivity

Step	**Branching Action**	**Alternate**
1	Offline Mode: Report stored locally and queued for sync.	
2	Template Error: Prompt user to reload or select another template.	

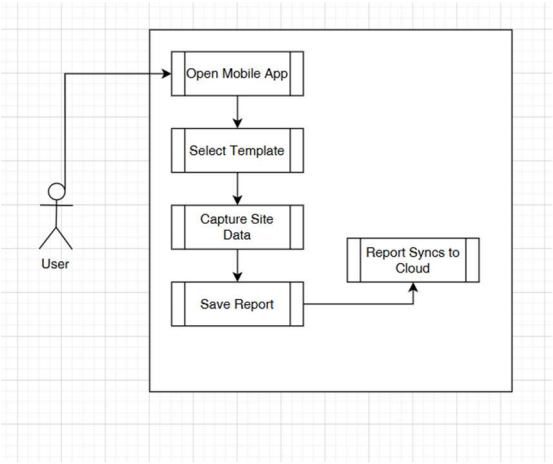

Figure 4 User Fills Report on site

3.5.4 User Completes the Report in the Web Application

Finalizing reports in the web application enables thorough review and User can trigger AI analysis, supporting proactive risk detection and informed decision-making.

USE CASE	3	Project Manager completes Report on Web App and Submits for AI analysis
Description of Goal in Context	Enable the Project Manager to finalize a completed report on the web application and trigger AI analysis for risk detection, delays, and material shortages.	
Preconditions	Report has been created and synced from the mobile app. Manager is authenticated in the web application. AI engine is operational and accessible.	
Post Conditions, Success End Condition	Report finalized and locked for analysis. AI engine processes report and displays flagged issues on dashboard. Confirmation message displayed to user.	
DESCRIPTION of Scenario	The Project Manager logs into the web application, reviews the synced report from the mobile app, makes any necessary edits, and finalizes it. They then trigger AI analysis, which scans the report text and images for risks, delays, and shortages. Results are displayed on the dashboard for review.	

Main Flow

Step	Action	Response
1	Log into web application	Authentication verified
2	Open synced report	Display report details
3	Review and edit report	Validate changes
4	Finalize report	Lock report for analysis
5	Trigger AI analysis	Show progress indicator
6	Display flagged issues	Update dashboard with results

EXCEPTIONS or ERROR Flow
Description

Step	Branching Action	Alternate
1	AI engine fails	Display error and allow retry.
	Report validation error	Prompt correction before analysis.

ALTERNATIVE or VARIATION Flow
Description
Variations in how the report is submitted depending on connectivity

Step	Branching Action	Alternate
1	Manager cancels AI analysis	Report remains editable.
2	AI analysis delayed due to server load	Notify user and queue task.

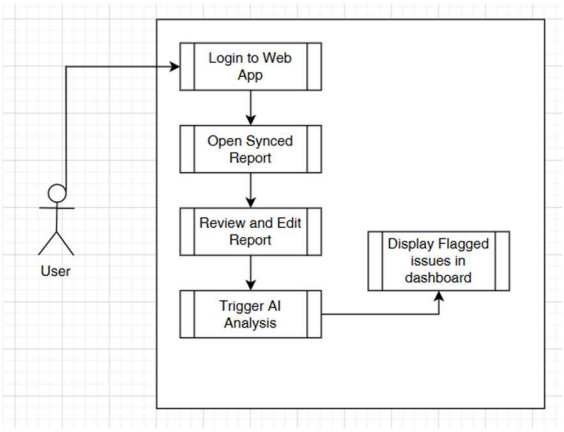

Figure 5 User completes report in web app

3.5.5 Use Case Justification

The selected use cases were chosen because they represent core workflows in construction reporting and directly address the pain points identified during stakeholder analysis and requirements elicitation. Each use case aligns with the system's objectives of improving efficiency, accuracy, and compliance:

- **User Creates Report Template** – Ensures standardisation and regulatory compliance by allowing managers to design structured templates tailored to project needs. This addresses the lack of consistency and manual template creation highlighted as a major pain point.
- **User Fills Report On-Site** – Captures real-time site data using predefined templates, reducing delays and miscommunication caused by manual notetaking and image compilation. Offline-first capability ensures reliability in low-connectivity environments.
- **User Completes Report in Web Application** – Enables thorough review and finalisation of reports, incorporating AI-driven analysis for risk detection and material shortage identification. This supports proactive decision making and improves overall reporting quality.

These use cases were prioritised because they form the backbone of the reporting workflow: template creation → data capture → report finalisation and analysis. Together, they ensure that ReportBuilderPro delivers a streamlined, user-centric process that addresses industry

challenges such as fragmented communication, compliance risks, and inefficient manual workflows.

3.6 Preliminary High-level Architecture

The architecture of ReportBuilderPro will be designed to support offline-first reporting, secure data handling, and AI-driven analysis, ensuring reliability in construction environments with intermittent connectivity.

The system will consist of three main layers frontend, backend, and storage. The frontend will include a web application built with React and a mobile application built with React Native. These will provide user interfaces for creating templates, capturing site data, and managing reports. Authentication will be handled through Azure Active Directory to ensure secure access and role-based permissions.

The mobile application will support offline caching so that data can be captured without connectivity. A syncing engine will upload data to the backend when connectivity is restored. The backend will be developed using FastAPI and Node.js to manage business logic and integrate with the AI/NLP engine. The AI engine will use spaCy for text preprocessing and Scikit-learn for classification tasks such as detecting risks, delays, and material shortages.

Data will be stored in MongoDB for structured and semi-structured report content, while AWS S3 will be used for storing images and PDF reports. Once reports are synced, the AI issue scanner will process them and flag potential issues, which will be displayed on the dashboard for review. Reports will then be exported as PDFs for compliance and sharing.

This architecture will ensure security, and responsiveness while addressing key challenges such as intermittent connectivity and the need for intelligent reporting.

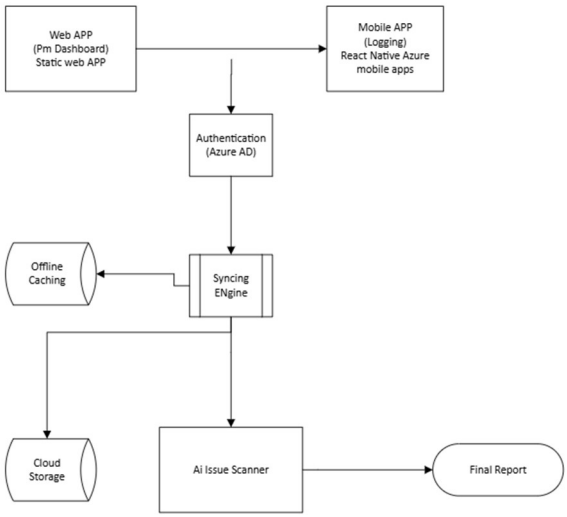

Figure 6. ReportBuilderPro System Workflow

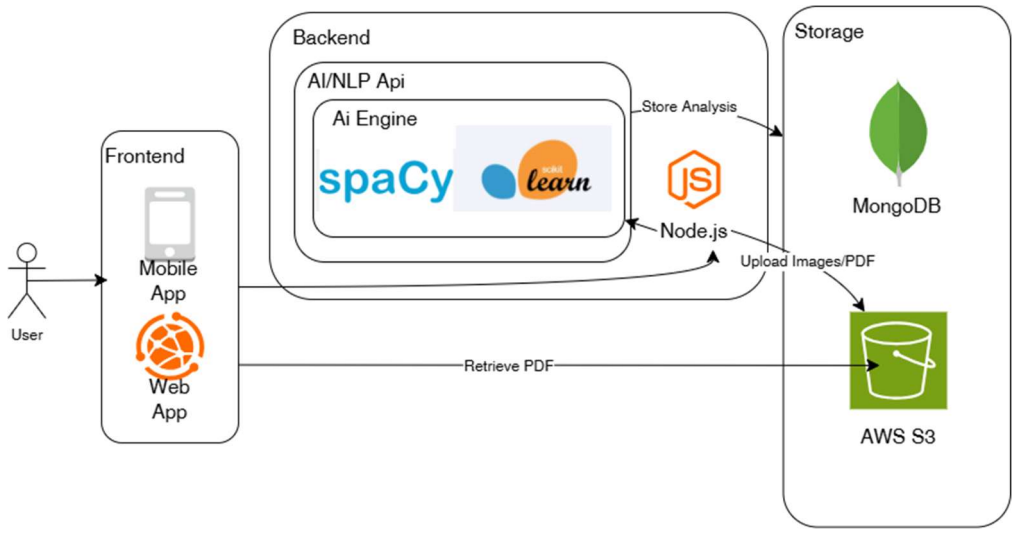

Figure 7 ReportBuilderPro System ArchitectureS

3.7 Key Components Breakdown

This section outlines the major subsystems of ReportBuilderPro, emphasizing their roles, technologies and how they interconnect to support seamless site reporting and project oversight.

1. Mobile Application

Purpose: Enables field operatives to capture and submit reports directly from construction sites, even in low-connectivity environments.

Core Features Include

Offline First Reporting

- Local data storage with deferred synchronisation.
- Justification: Ensures uninterrupted data capture in remote or signal-poor areas.

Image Capture

- Multimedia inputs enhance report richness and context.
- Justification: Supports visual documentation and hands-free reporting.

Backend Synchronisation

- Automatic sync when connectivity is restored.
- Justification: Maintains data integrity and continuity across devices.

2. Web Application

Purpose: Provides administrative and analytical capabilities for stakeholders. Acts as the central hub for ReportBuilderPro

Core Features Include

Template Builder

- Customizable report templates for different project types.
- Justification: Promotes consistency and adaptability across use cases.

Project Dashboard

- Visual tracking of report metrics, issues, and progress.
- Justification: Facilitates data-driven decision-making and oversight.

PDF Report Generation

- Exportable, standardised reports for sharing and compliance.
- Justification: Aligns with industry documentation practices.

3. Backend & API Layer

Purpose: Serves as the system's data and logic core, enabling secure storage, processing, and integration.

Technologies:

MongoDB

- NoSQL database for flexible document storage.
- Justification: Suited for unstructured and semi-structured report data.

FastAPI

- Lightweight Python-based framework for API development.
- Justification: Enables rapid development and seamless integration with AI modules.

AWS (Amazon Web Services)

- Cloud-based file storage and processing.
- Justification: Ensures scalability, reliability, and secure file handling.

4. AI/NLP Feature

Purpose: Enhances report intelligence through automated text analysis and risk detection.

Technologies and Methods:

Functional Requirements

- The system shall provide automated text analysis of report content using NLP techniques.
- The system shall classify report text into categories such as delays, risks, and material shortages.
- The system shall flag identified issues and display them on the dashboard for user review.
- The system shall allow users to trigger AI analysis manually after report completion.

Non-Functional Requirements

- NLP processing shall complete within 3 seconds for standard reports to ensure responsiveness.
- The AI engine shall achieve at least 85% accuracy in classification tasks based on validation datasets.
- The system shall provide clear feedback when AI analysis fails or is delayed due to server load.

- AI outputs shall be advisory, not authoritative, and clearly marked as system-generated insights.

Justification These requirements ensure that the AI/NLP feature enhances reporting intelligence without compromising usability or transparency. They address stakeholder needs for proactive risk detection and align with industry trends in intelligent document processing.

3.8 User Interface Mock-ups

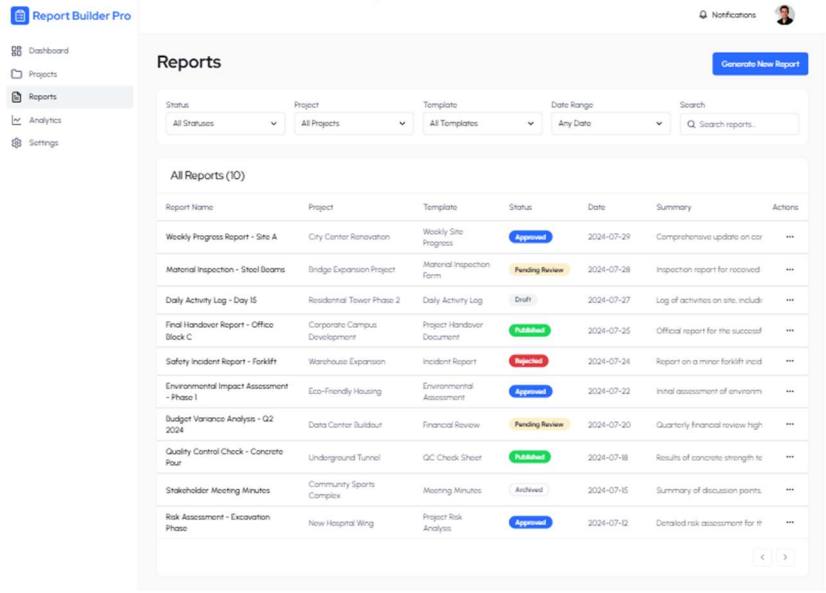

Figure 8. Sample Ui Mock-up 1

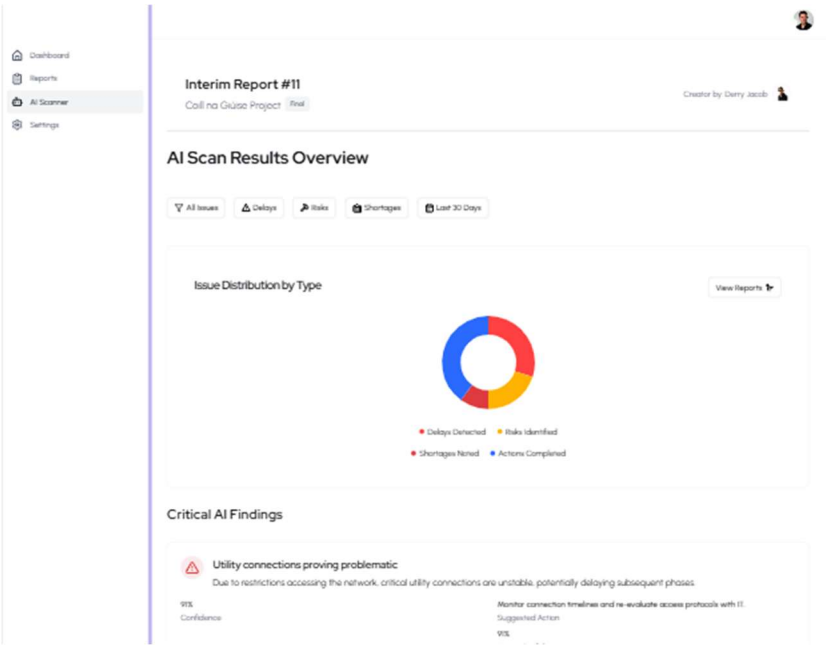

Figure 9.Sample Ui Mock-up 2

3.10 UX and Accessibility Requirements

The UX design of Report Builder Pro is central to its success, particularly given its role in facilitating structured reporting in dynamic, field based environments. The application is designed with a human centred approach, prioritizing the needs, behaviours, and constraints of its end users namely site engineers, project managers, and safety officers.

To ensure relevance and usability, user research was conducted (see Section 2) Aswell as clearly defined use cases (see Section 3.6), focusing on pain points in current reporting workflows, such as time-consuming data entry, lack of mobile responsiveness, and difficulty accessing summarised insights. Future iterations will incorporate user testing to evaluate prototype performance and refine features based on direct feedback.

The design approach is grounded in established usability and accessibility principles, ensuring that Report Builder Pro delivers a seamless, efficient, and inclusive reporting experience in high-pressure field conditions.

3.10.1 Shneiderman's Eight Golden Rules of Interface Design [22]

These rules guide the interface design of Report Builder Pro, ensuring that users can report quickly, accurately, and confidently in the field:

- **Strive for Consistency**
  Report templates, navigation patterns, and UI components are consistent across mobile and desktop, reducing training time and supporting cross-platform use on-site and in the office.
- **Enable Universal Usability**
  The platform supports a range of devices and user capabilities, including rugged tablets and low-bandwidth environments, ensuring accessibility for all field roles.
- **Offer Informative Feedback**
  Real-time feedback (e.g., sync status, submission confirmations) helps users understand system responses, critical for time-sensitive reporting.
- **Design Dialogs to Yield Closure**
  Reporting workflows are structured to guide users to completion, with clear summaries and confirmations that reinforce task success.
- **Prevent Errors**
  Smart form validation and contextual prompts reduce the risk of incorrect data entry, which is especially important when reporting under pressure or in poor conditions.

- **Permit Easy Reversal of Actions**
  Users can edit or undo entries, reset filters, and revise reports—supporting flexibility and reducing anxiety around mistakes.
- **Support Internal Locus of Control**
  Users initiate and control reporting actions, from choosing templates to customizing summaries, reinforcing confidence and autonomy.
- **Reduce Short-Term Memory Load**
  Frequently used tags, categories, and templates are surfaced contextually, allowing users to focus on the task rather than remembering system details.

3.10.2. ISO Dialogue Principles [21]

Report Builder Pro also aligns with the ISO 9241-110 standard for dialogue principles, ensuring usability and accessibility in professional environments

- **Suitability for the Task**
  Interfaces are tailored to construction reporting tasks, using industry-specific terminology and workflows (e.g., “Safety Issue,” “Site Condition”).
- **Self-Descriptiveness**
  Each screen and input field clearly communicates its function, helping users understand what to do without needing external guidance.
- **Controllability**
  Users can navigate freely, pause and resume reporting, and control data inputs critical for field users who may be interrupted or multitasking.
- **Conformity with User Expectations**
  The design reflects familiar patterns from construction and safety reporting tools, reducing friction and improving adoption.
- **Error Tolerance**
  The system supports recovery from mistakes with clear error messages and suggestions (e.g., “Please enter a valid location”), minimizing disruption.
- **Suitability for Individualisation**
  Users can adjust layout, font size, and themes to suit their working conditions, such as bright outdoor environments or low-light indoor settings.
- **Suitability for Learning**
  Embedded guidance, tooltips, and onboarding flows help new users get started quickly, while experienced users can skip or customize their experience.

3.10.3. Accessibility and ARC Toolkit Integration

To ensure inclusivity and compliance with modern standards, Report Builder Pro integrates accessibility testing using the ARC Toolkit, and follows WCAG 2.2 [23] guidelines.

- **Colour Contrast Testing**
  The UI colour palette is tested to meet contrast ratios for readability, especially in outdoor lighting conditions.

- **Keyboard Navigation**
  All features are accessible via keyboard, supporting users with motor impairments and improving accessibility on rugged field devices.
- **Customizable UI**
  Users can adjust text size, switch to high-contrast themes, and enable simplified layouts for better visibility and focus.
- **Screen Reader Compatibility**
  Semantic HTML and ARIA labels are used to ensure compatibility with assistive technologies.
- **Touch-Friendly Design**
  Buttons and input fields are sized appropriately for touch interaction, reducing errors and improving usability on mobile devices.

3.11 Error message and Feedback Design

ReportBuilderPro will prioritize clear, actionable feedback such as:

• **Error Messages**

Tailored messages like “Please enter a valid format image i.e. png or jpg” will guide users to correct inputs.

• **Success Feedback**

The creation of subtle animations and checkmarks will confirm completed actions, such as saving a pdf or AI analysis completion.

• **Responsiveness**

Smooth transitions and quick load times will reinforce a sense of system reliability and efficiency to users.

3.12 Conclusions

This chapter outlined the system analysis for ReportBuilderPro, covering everything from stakeholder requirements and functional specifications to system architecture, UI mock-ups, and technology choices. It established a clear, user-centred foundation for development, guided by Feature-Driven Development, accessibility standards, and proven UX principles like Schneiderman’s Golden Rules[22] and ISO 9241-110 [21]. Together, these elements ensure the platform is intuitive, scalable, and purpose-built for efficient, cross-device reporting in demanding field environments.

4. System Design

4.1 Introduction

This section introduces the approach taken to transform the requirements outlined in Section 3 (System Analysis) into a structured and system design for ReportBuilderPro. The goal is to ensure that the system architecture, components, and workflows are aligned with the practical needs of field operatives and project stakeholders in the construction industry.

The design process is guided by a combination of:

- **Feature-Driven Development (FDD)** to support iterative delivery and continuous feedback.
- **User-centred design principles** to ensure usability, accessibility, and relevance in real-world site conditions.
- **Modular architecture** to enable flexibility, maintainability, and future scalability.
- **Best practices in cloud infrastructure and AI integration**, ensuring the system is robust, secure, and capable of intelligent reporting.

This chapter outlines the chosen software methodology, the logical and physical architecture of the system, the design strategy applied to each component, and considerations for deployment and production environments. Together, these elements form the blueprint for building a reliable and intelligent site reporting platform.

4.2 Software Development Methodology

To guide the development of ReportBuilderPro, a Feature-Driven Development (FDD) approach was adopted. FDD is a client-centric, iterative methodology that focuses on designing and building software around clearly defined, user-valued features. This approach aligns strongly with the goals of ReportBuilderPro, which prioritises usability, modularity, and responsiveness to real-world site reporting needs.

As outlined by Scott W. Ambler in *"Feature Driven Development (FDD) and Agile Modeling"*[15], FDD is built on five core activities:

Develop an Overall Model

- Core entities such as users, reports, templates, risk flags, and workflows were mapped out.
- This model was informed by expert elicitation using the IDEA Protocol [14] (see Section 2.4.1).

Build a Feature List

- A comprehensive list of desired features was compiled based on stakeholder input and workflow analysis.

Plan by Feature

- Features were prioritised based on user feedback, feasibility, and impact.

- Logical grouping into modules (e.g., reporting, dashboard, risk analysis) supported structured planning.

Design by Feature

- Each feature was designed with a focus on simplicity, accessibility, and relevance to field workflows.
- UX principles such as Shneiderman's Golden Rules [22] and ISO 9241-110 [21] were applied to ensure intuitive interaction.

Build by Feature

- The system architecture supports modular, feature-based development, enabling iterative delivery and continuous refinement.

FDD Principles Applied to ReportBuilderPro

- **Feature-Centric**
  Each feature represents a small, client-recognisable unit of functionality (e.g., "Submit Site Report," "Flag Risk," "Generate PDF").
- **Iterative and Incremental**
  Features are developed in short cycles, allowing for ongoing feedback and adaptation to evolving site needs.
- **Domain-Driven**
  The system is modelled around real-world entities and processes in construction reporting, ensuring relevance and clarity.
- **Team-Oriented**
  Roles such as Feature Owner and Class Owner (as described by Ambler [8]) promote accountability and collaboration across design and development.

As Ambler states, *"FDD blends several industry-recognized best practices into a cohesive whole. These practices are driven from a client-valued functionality (feature) perspective."* [8] This makes FDD particularly well-suited to ReportBuilderPro, where stakeholder engagement and field usability are central to success.

4.2.1 Comparison between FDD and other methodologies

While Agile and Scrum are also widely adopted methodologies in software development, Feature-Driven Development (FDD) was selected for ReportBuilderPro due to its stronger alignment with the project's structure and goals.

Comparative Analysis of Development Methodologies

Methodology	Characteristics	Relevance to ReportBuilderPro
FDD	- Feature-centric - Structured around domain modelling and feature lists - Focuses on designing and building by feature	FDD provides a clear path from requirements to implementation, making it ideal for a system like ReportBuilderPro that is built around user-recognisable reporting features.
Agile [24]	- Values individuals, collaboration, and adaptability - Emphasizes working software and customer feedback	Agile principles underpin the overall development philosophy, especially in terms of iterative progress and responsiveness to user needs.
Scrum[25]	- Framework within Agile - Uses sprints, roles (Scrum Master, Product Owner), and ceremonies (stand-ups, retrospectives)	Scrum offers strong team coordination, but its sprint-based structure may be less suited to the modular, feature-first nature of this project.

Development Methodology Decision Matrix – ReportBuilderPro

To determine the most suitable software methodology for ReportBuilderPro, a decision matrix was developed using criteria directly aligned with ReportBuilderPro's goals and constraints. These criteria included user-centric design, modular feature delivery, domain modelling support, iterative development, stakeholder engagement, and scalability for complex systems.

Each methodology i.e. FDD, Agile, and Scrum was evaluated against these factors based on its core principles and practical fit for a feature-first, field-oriented reporting platform. Feature-Driven Development (FDD) emerged as the most appropriate choice due to its structured approach to building client-recognizable features, its emphasis on domain modelling, and its compatibility with modular system architecture.

While Agile and Scrum offer valuable practices, FDD provides the clearest path from stakeholder requirements to implementable features, making it the best fit for ReportBuilderPro's development strategy.

Criteria	FDD	Agile	Scrum
User-Centric Design	Yes	Yes	Yes
Modular Feature Delivery	Yes	Partial	Partial
Domain Modeling Support	Yes	No	No
Iterative Development	Yes	Yes	Yes
Team Roles & Accountability	Yes	Yes	Yes
Stakeholder Engagement	Yes	Yes	Yes
Suitability for ReportBuilderPro	Yes	No	No

4.3 High Level System Architecture

This section presents the high level architecture and infrastructure of ReportBuilderPro, integrating insights from earlier analysis and design decisions. It outlines the logical structure of the system, the initial physical infrastructure supporting development and testing, and considerations for deployment in a production environment.

4.3.1 Logical Architecture

The logical architecture of ReportBuilderPro will be modular and feature-driven, reflecting the principles outlined in Section 4.2 (Software Methodology) and the component breakdown in Section 3.5. Will be composed of four interconnected subsystems.

1. Mobile Application

The mobile application will be developed to support on-site data capture, particularly in environments with limited connectivity. It will

- Operate on an offline-first basis, storing data locally and syncing when a connection is available.
- Allow users to upload images, dictate notes, and complete structured checklists.
- Be tailored for use by field operatives such as project managers and quantity surveyors.

2. Web Application

The web application will serve as the central hub for administrative tasks. Template creation and report refinement. It will

- Enable users to create and customize report templates.
- Display flagged risks, delays, and material shortages via a visual dashboard.
- Support PDF generation for client-facing documentation.

3. Backend & API Layer
The backend will act as the system's core for data processing and communication. It will:

- Manage user authentication and role-based access control.
- Handle data storage, file uploads, and report submissions.
- Provide API endpoints to facilitate communication between the mobile and web applications.

4. AI/NLP Engine
An integrated AI engine will enhance the system's intelligence by analysing report content. It will:

- Use Natural Language Processing (NLP) to scan text for risks, delays, and shortages.
- Apply machine learning models to classify and flag issues.
- Continuously improve through feedback and usage patterns.

4.3.2 Physical Infrastructure

The initial infrastructure will be cloud-based to support rapid development, secure testing, and modular deployment.

- **Hosting**: The web application and backend services will be hosted on Azure, selected for its integration with development tools and scalability.
- **Storage**: Uploaded images and generated PDFs will be stored securely using AWS S3. MongoDB Atlas will be used for flexible, document-oriented data storage.
- **Networking**: All data will be transmitted over HTTPS, with secure access protocols in place to protect user information.

4.3.3 Deployment and Production Environment

Although full-scale commercial deployment is outside the scope of the current phase, the system will be designed with production-readiness in mind.

- **Environment Setup**: Separate development, testing, and production environments will be established to support iterative development and quality assurance.
- **Deployment Strategy**: CI/CD pipelines will be implemented to automate deployment and reduce manual errors.
- **Security and Compliance**: The system will comply with GDPR standards and include encryption for data at rest and in transit.
- **Monitoring and Maintenance**: Logging and alerting mechanisms will be used to monitor system health and performance, with modular architecture supporting future scalability.

4.4 Component Level Design

The design of ReportBuilderPro is guided by the principles of Feature-Driven Development (FDD), a methodology that emphasizes building software around clearly defined, user valued features. This approach ensures that the system is modular and aligned with real-world workflows in the construction industry.

4.4.1 Development Methodology Design Considerations

FDD is structured around five key activities:

1. **Develop an Overall Model** – Define core entities and relationships.
2. **Build a Feature List** – Identify user-valued features based on stakeholder input.
3. **Plan by Feature** – Prioritize and group features into logical modules.
4. **Design by Feature** – Create design specifications for each feature.
5. **Build by Feature** – Implement features iteratively and incrementally.

This methodology was selected for its alignment with the modular architecture of ReportBuilderPro and its ability to support continuous refinement based on user feedback.

4.4.2 Module Design

The system will be designed around the following core modules, each corresponding to a set of prioritised features:

1. Reporting Module

Features

- Create and customize report templates.
- Capture structured data on-site via mobile.
- Submit reports for AI analysis.

Design Considerations

- Templates will be built using reusable UI components (React).
- Mobile forms will support offline-first data entry.
- Reports will be stored in MongoDB with flexible schema support.

2. Dashboard & Analytics Module

Features

- Visualize flagged risks, delays, and material shortages.
- Display AI-generated insights and recommendations.

Design Considerations

- Dashboard will use dynamic data binding and filtering.
- AI outputs will be integrated via FastAPI endpoints.
- Icons and visual cues will support quick decision-making.

3. AI/NLP Module

Features

- Scan report text for risks and delays.
- Classify issues using trained ML models.
- Improve accuracy through continuous learning.

Design Considerations

- NLP pipeline will use spaCy for preprocessing.
- Classification models will be built with Scikit-learn.
- Feedback loops will be designed to refine model performance.

4. PDF Generation Module

Features

- Export reports in professional PDF format.
- Include AI insights and visual summaries.

Design Considerations

- HTML templates will be converted using pdfkit.
- Layouts will be optimized for readability and compliance.

5. Authentication & Access Control Module

Features

- Secure login and role-based access.
- Different permissions for mobile and web users.

Design Considerations

- Authentication will be handled via Node.js backend.
- Access rules will be enforced at API and UI levels.

The design system will follow key principles to ensure usability.

- **Modularity** Each feature will be encapsulated in its own component or service.
- **Scalability** The architecture will support future expansion (e.g., integration with BIM or ERP systems).
- **Accessibility** UI components will be tested using ARC Toolkit and follow WCAG 2.2 guidelines.[22]
- **Consistency** Design patterns will be reused across mobile and web interfaces.
- **Error Tolerance** Smart validation and feedback mechanisms will reduce user errors.

4.4.3 Visual Design

The visual design will be guided by:

- **Material Design principles** for layout and interaction.
- **Responsive design** to ensure usability across devices.
- **Touch friendly components** for mobile field use.
- **High contrast themes** for outdoor visibility.

4.5 Data Design

To support the modular and feature-driven architecture of ReportBuilderPro, a conceptual Entity Relationship Diagram (ERD) has been developed. This ERD outlines the core entities within the system and the relationships between them, ensuring that the data model aligns with the reporting workflows and AI-enhanced analysis features described in earlier sections.

The ERD is designed to reflect the following principles:

- **Modularity** Each entity corresponds to a distinct feature or subsystem.
- **Clarity** Relationships are kept intuitive to support maintainability and ease of development.

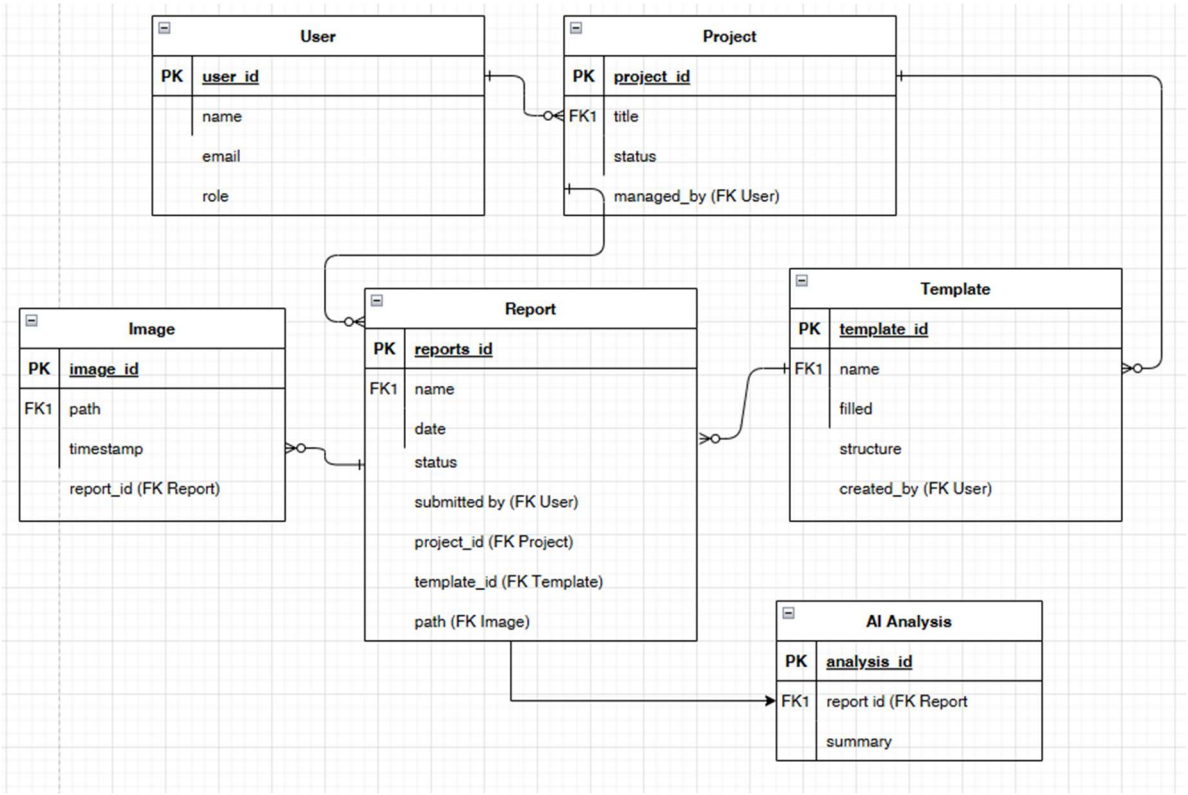

Figure 10 ReportBuilderPro Conceptual ERD

Document Data Model (MongoDB)

ReportBuilderPro will use a document-oriented approach for storing reports and related data in MongoDB. A sample structure:

Figure 11 Sample of data model

Data Validation Rules

- **Required Fields:** report_id, project_id, submitted_by, date, status.
- **Status Values:** Must be one of draft, submitted, completed.
- **Image Path:** Must follow AWS S3 URL format.
- **AI Analysis Flags:** Must match predefined categories (risk, delay, material_shortage).
- **Date Format:** ISO 8601 standard for consistency.

4.6 Feature to Component mapping

The table below demonstrates how each feature in ReportBuilderPro is supported by a specific system component, reinforcing the modular design and ensuring traceability between functional requirements and technical implementation.

Feature	System Component	Technology	Design Note
Submit Site Report	Mobile application	React	Offline first design with local caching and deferred sync
Create and Edit Report template	Web Application	React.js	Drap and drop interface for

			template customisation
Upload Images and notes	Mobile Application	React Native, AWS S3	Support multimedia inputs and sync when online
AI Scan	NLP Engine	spaCy, Scikit-learn, FastAPI	NLP pipeline for text analysis and classification of issues
Generate PDF Report	Backend	Pdfkit	Convert structure format to professional PDF
View Dashboard and Insights	Web Application	React.js, FastAPI	Display flagged issues in real time
User Login and Access Control	Backend	Node.js, MongoDB	Role based secure authentication
Store and Retrieve Data	Backend	MongoDB	Flexible schema to support varied reports
Sync Mobile and Web Data	Backend	FastAPI , Node.js	Ensure seamless data flow between mobile and web

4.7 Outline of ReportBuilderPro Tech Stack

ReportBuilderPro Tech Stack

Frontend	React/React Native
Preprocessing	spaCy
Model	Scikit-learn
Api	FastAPI
Database	Mongodb
Pdf export	pdfkit
Hosting	AWS
NLP Processing	spaCy
ML Models	Scikit-learn
Accessibility Testing	ARC Toolkit

4.8 Natural Language Processing (NLP) Design

Natural Language Processing will play a key role in ReportBuilderPro's intelligent reporting capabilities. Unlike tradition reporting systems such as SiteCam and Fieldwire, RportBUilderPro will leverage NLP to automatically interpret unstructured text and extract actionable insights. These identifying risks, delays and material shortages directly from the user submitted reports.

The inclusion of NLP is just a backend enhancement; it's a core feature that differentiates ReportBuilderPro and supports multiple user facing functionalities. From flagging issues in the dashboard to generating AI enhanced summaries for PDF exports, NLP is tightly integrated into the system's architecture and aligns with the Feature-Driven Development (FDD) methodology adopted for this project.

Recent literature emphasizes that NLP systems must be designed not only for performance, but also with ethical integrity in mind. Leidner and Plachouras (2017) [23] argue for an "ethical by design" approach, where ethical considerations are embedded throughout the entire lifecycle of NLP development, from data collection and model training to deployment and user interaction. This is particularly relevant for ReportBuilderPro, which processes sensitive site data and generates automated insights. To align with these principles, the system's NLP engine will be developed using transparent methods, trained on domain specific data, and evaluated not only for accuracy but also for fairness, privacy, and user impact.

Design Overview

The NLP engine will be structured as a modular pipeline, enabling flexibility, maintainability, and continuous improvement.

Preprocessing

- Tokenization, sentence segmentation, and entity recognition using **spaCy**.
- Cleaning and normalization of report text to reduce noise and improve model input quality.

Classification

- Use of Scikit-learn to train supervised models that classify report content into categories such as:
- Risk
- Delay
- Material Shortage
- Models will be trained on annotated construction report data and refined through iterative feedback.

Integration

- The NLP engine will be exposed via **FastAPI**, allowing seamless communication with the dashboard and backend.
- Flagged issues will be returned as structured data for display in the dashboard and inclusion in PDF summaries.

Output:

- Each report will be enhanced with a list of flagged items, confidence scores, and suggested actions

NLP Pipeline Overview

Stage	Description	Technology/Method
Text Access	Retrieve report text from the database or file storage.	MongoDB, AWS S3
Preprocessing	Clean and prepare text for analysis: tokenization, lemmatization, etc.	spaCy
Feature Extraction	Convert text into numerical features for model input	Scikit-learn, spaCy
Classification	Apply trained models to classify content into categories	Scikit-learn
Flag Generation	Generate structured flags with labels and confidence scores.	Custom Logic
Dashboard Display	Send flagged issues to the frontend for visualization and user review.	FastAPI, React.js
Feedback Loop	Capture user feedback on flagged items to improve model accuracy over time.	Backend logging
PDF Integration	Embed NLP insights into the final report output.	pdfkit

4.9 User Interface Design

The user interface (UI) will be designed to provide a clear, intuitive, and efficient experience for all users, ensuring that tasks such as reporting, reviewing, and managing data will be completed with minimal effort and error. The design approach will prioritise usability, accessibility, and responsiveness across devices.

Design Goals

- **Clarity and Consistency:** The system will maintain uniform layouts, terminology, and visual hierarchy across all screens.
- **Error Prevention and Recovery:** Real-time validation, mandatory field indicators, and clear error messages will be incorporated.
- **Reduced Cognitive Load:** Logical grouping of fields, progressive disclosure for advanced options, and familiar interaction patterns will be applied.
- **Accessibility:** The interface will comply with WCAG standards for colour contrast, keyboard navigation, and screen reader compatibility.
- **Responsiveness:** Layouts will be optimised for desktop, tablet, and mobile devices.

Key Components

1. **Navigation**
    - A persistent top-level menu will provide access to core modules (Dashboard, Reports, Settings).
    - Search and filter options will enable quick access to data.
2. **Reporting Forms**
    - **Consistency:** Field labels and input types will be standardised across all report templates.
    - **Error Prevention:** Inline validation will check mandatory fields and formats before submission.
    - **Controllability:** Users will be able to edit drafts, confirm submissions, and undo recent actions.
    - **Feedback:** Success and error messages will be displayed prominently, following ISO dialogue principles.
3. **Dashboards**
    - Visual summaries of key metrics will be presented using charts and tables.
    - Interactive filters will allow users to customise views.
    - Export options will support PDF and Excel formats.
4. **Feedback and Notifications**
    - Real-time status updates will inform users of submitted reports.
    - Alerts will highlight incomplete or overdue tasks.
    - Non-intrusive notifications will communicate system updates.

The user interface design will incorporate established usability principles and ISO dialogue guidelines to ensure efficiency, clarity, and accessibility across all platforms. To satisfy Shneiderman's rule of reducing short-term memory load, frequently used report fields and template components will be positioned at the top of the interface, supported by auto-complete suggestions. For dashboards, Shneiderman's informative feedback rule will guide the use of colour-coded alerts and status messages to communicate system states, flag risks, and provide timely updates. In mobile design, the ISO principle of suitability for the task will be applied by prioritising large, touch-friendly controls tailored for operators working in outdoor, time-sensitive environments. Collectively, these design decisions will create an interface that is intuitive, responsive, and aligned with the operational needs of its users.

4.10 Conclusions

The system design for ReportBuilderPro presents a clear and cohesive blueprint for delivering a structured, intelligent reporting platform tailored to the needs of construction professionals. Built around a modular architecture and guided by Feature Driven Development (FDD), the design ensures that each component directly supports user valued functionality, from mobile data capture to AI-enhanced analysis.

By aligning the design with real workflows and usability standards such as Shneiderman's Golden Rules and ISO 9241-110, the system is positioned to deliver a seamless and accessible experience across devices and user roles. The integration of AI and NLP not only differentiates ReportBuilderPro from existing tools like SiteCam and Fieldwire but also empowers users with proactive insights and intelligent reporting capabilities.

The design decisions made in this chapter lay the foundation for a scalable, maintainable, and user-centric system. These choices will guide the development and testing phases, ensuring that the final product remains aligned with stakeholder needs and industry best practices.

5. Testing and Evaluation

5.1 Introduction

To ensure the reliability and effectiveness of Report Builder Pro, a structured testing and evaluation process is essential. This process verifies that the system performs as intended, meets user requirements, and adheres to the design principles established during development.

This chapter outlines how Report Builder Pro will be tested and evaluated to provide a clear overview of the system's performance and usability. Testing and evaluation will ensure that Report Builder Pro delivers accurate reporting, intuitive user interactions, and reliable performance across all environments. The approach will reflect the system's logical architecture, which includes:

- **Data Layer** Handles input validation, storage, and retrieval of reporting data.
- **Application Layer** Manages report generation, AI summarisation, and workflow logic.
- **Presentation Layer** Provides dashboards, reporting forms, and mobile interfaces.

Each layer will undergo targeted testing to confirm functionality, usability, and compliance with design principles outlined in Section 3 and Section 4.

5.2 Plan for Testing

5.2.1 Reporting Workflow Testing

Objective: Verify that users can create, edit, and submit reports without errors.

Approach: Simulate real-world reporting tasks using predefined templates.

Tests:

- Mandatory field validation.
- Auto-complete suggestions for frequently used fields (Shneiderman's memory load principle).
- Error handling for incomplete or invalid data.

5.2.2 Dashboard Testing

Objective: Ensure dashboards provide accurate, real-time feedback.
Approach: Test colour-coded alerts, status messages, and risk indicators.
Tests:

- Correct display of alerts for overdue reports.
- Real-time updates when data changes.
- Responsiveness across desktop and mobile

5.2.3 Mobile Interface

- **Objective:** Validate usability in outdoor, time-sensitive environments.
- **Approach:** Field testing with touch-friendly controls and simplified workflows.
- **Tests:**
    - Button size and spacing for touch accuracy.
    - Offline mode behaviour.
    - Performance under limited connectivity

Overall Test Plan

Component	Scenario	Expected Result
Reporting Form	Submit with missing mandatory fields	Inline error message displayed
Dashboard Alerts	Overdue report detected	Colour-coded alert shown immediately
Mobile UI	Operator creates report outdoors	Smooth navigation, no input errors

5.3 Plan for Evaluation

Evaluation will focus on usability, efficiency, and compliance with ISO and Shneiderman principles

- **Usability Testing:** Observe users completing reporting tasks; measure time, error rate, and satisfaction.
- **Heuristic Review:** Apply Shneiderman's rules (e.g., informative feedback, memory load reduction).
- **Accessibility Audit:** Check WCAG compliance for colour contrast, keyboard navigation, and screen reader support.
- **Performance Metrics:** Measure report generation speed and dashboard refresh rates under typical and peak loads.

5.4 Conclusions

Testing and evaluation will confirm that Report Builder Pro meets its functional and usability objectives. By aligning tests with the logical architecture and design principles, the system will deliver accurate reporting, clear feedback, and mobile-friendly workflows. Insights from evaluation will inform iterative improvements, ensuring the platform remains robust and user centred.

6. System Prototype

As least 2 pages, but as many as you like (with lots of code samples).

6.1 Introduction

Outline of approach to developing the prototype
Identify frameworks used for any part of the system

6.2 Prototype Development

Develop Code for each part of system (Logical Architecture)

6.3 Results

Perform tests identified in section 5

6.4 Evaluation

Evaluate results
Use evaluation approaches from Section 5 to assess the outcomes of the development

6.5 Conclusions

7. Issues and Future Work

7.1 Introduction

Purpose of the section

7.2 Issues and Risks

Describe aspects of the project that had different outcomes to the expected outcomes, based on evaluation of results

Identify key elements of uncertainty that will lead to risks in completing the project

7.3 Plans and Future Work

Consider and briefly describe how to address any risks and how to complete the system

Consider and describe how to complete the system based on the available time

7.3.1 Project Plan with GANTT Chart

Based on the future plans, describe how to complete the project

Break down the plan into a detailed yet realistic schedule.

Present a Gantt chart to summarise the project plan

References

[1] ‘Spotlight on Build Digital: Driving Innovation in Irish Construction and Built Environment Industries’, gov.ie. Accessed: Nov. 21, 2025. [Online]. Available: https://www.gov.ie/en/department-of-public-expenditure-infrastructure-public-service-reform-and-digitalisation/spotlight-on-build-digital-driving-innovation-in-irish-construction-and-built-environment-industries/

[2] ‘(PDF) The Challenges of Standardization of Products and Processes in Construction’, in *ResearchGate*, doi: 10.13140/2.1.3993.7600.

[3] D. Shamsollahi, ‘Construction Progress Monitoring and Reporting using Digital Images and Computer Vision Techniques - A Review’. [Online]. Available: https://www.iaarc.org/publications/2022_proceedings_of_the_39th_isarc_bogota_colombia/construction_progress_monitoring_and_reporting_using_digital_images_and_computer_vision_techniques_a_review.html

[4] Y. Adebayo, P. Udoh, X. B. Kamudyariwa, and O. A. Osobajo, ‘Artificial Intelligence in Construction Project Management: A Structured Literature Review of Its Evolution in Application and Future Trends’, *Digital*, vol. 5, no. 3, p. 26, Sept. 2025, doi: 10.3390/digital5030026.

[5] ‘Layout 1’. Accessed: Nov. 21, 2025. [Online]. Available: https://scsi.ie/wp-content/uploads/2025/08/SurveyorsJournal_August2025_web.pdf

[6] S. of C. S. Ireland, ‘Will AI help quantity surveying?’, *Surv. J.*, vol. 25, no. 3, pp. 20–22, Aug. 2025.

[7] D. Vararean-Cochisa and E.-L. Crisan, ‘The digital transformation of the construction industry: a review’, *IIM Ranchi J. Manag. Stud.*, vol. 4, no. 1, pp. 3–16, Jan. 2025, doi: 10.1108/IRJMS-04-2024-0035.

[8] E. Ireland, ‘EY report highlights critical role of innovation to ensure a productive and sustainable construction sector in Ireland’, EY report highlights critical role of innovation to ensure a productive and sustainable construction sector in Ireland. [Online]. Available: https://www.ey.com/en_ie/newsroom/2023/02/ey-report-highlights-critical-role-of-innovation-to-ensure-a-productive-and-sustainable-construction-sector-in-ireland

[9] ‘Home’, The National Planning Framework. Accessed: Nov. 21, 2025. [Online]. Available: https://www.npf.ie/

[10] ‘Government publishes updated National Development Plan’, gov.ie. Accessed: Nov. 21, 2025. [Online]. Available: https://www.gov.ie/en/department-of-the-taoiseach/press-releases/government-publishes-updated-national-development-plan/

[11] A. Naik, ‘The Front-End Dilemma: How to Choose the Perfect TECHNOLOGY for Your Application.’, *J. Comput. Sci. Technol. Stud.*, vol. 6, no. 1, pp. 211–216, Mar. 2024, doi: 10.32996/jcsts.2024.6.1.24.

[12] ‘The Invisible Technology How Backend Systems Shape Our Digital Lives | Request PDF’, ResearchGate. Accessed: Nov. 21, 2025. [Online]. Available: https://www.researchgate.net/publication/391980564_The_Invisible_Technology_How_Backend_Systems_Shape_Our_Digital_Lives

[13] M. Rathore and S. S. Bagui, ‘MongoDB: Meeting the Dynamic Needs of Modern Applications’, *Encyclopedia*, vol. 4, no. 4, pp. 1433–1453, Dec. 2024, doi: 10.3390/encyclopedia4040093.

[14] V. Hemming, M. A. Burgman, A. M. Hanea, M. F. McBride, and B. C. Wintle, ‘A practical guide to structured expert elicitation using the IDEA protocol’, *Methods Ecol. Evol.*, vol. 9, no. 1, pp. 169–180, 2018, doi: 10.1111/2041-210X.12857.

[15] ‘Feature Driven Development (FDD) and Agile Modeling’. Accessed: Nov. 03, 2025. [Online]. Available: https://agilemodeling.com/essays/fdd.htm

[16] ‘TU Dublin - Library Catalogue’. Accessed: Nov. 21, 2025. [Online]. Available: https://library.tudublin.ie/#
[17] J. Hew and M. Llorens Salvador, ‘Anseo! -: A modern approach to tracking student attendancein university Final project report’, TU Dublin, Dublin, 2023.
[18] B. McCarthy and J. Carswell, ‘Deep: Final project report’, TU Dublin, Dublin, 2023.
[19] S. Lewellen, ‘Identifying Key Stakeholders as Part of Requirements Elicitation in Software Ecosystems’, in *Proceedings of the 24th ACM International Systems and Software Product Line Conference - Volume B*, in SPLC ’20. New York, NY, USA: Association for Computing Machinery, Oct. 2020, pp. 88–95. doi: 10.1145/3382026.3431249.
[20] ‘(PDF) The Significance of Use Case Diagrams in Software Development Keywords Use Case Diagrams Software Development Requirements Elicitation System Design Stakeholder Communication Functional Requirements Visual Modeling Tools Iterative Development User-Centered Design Software Engineering Methodologies’, ResearchGate. Accessed: Nov. 14, 2025. [Online]. Available: https://www.researchgate.net/publication/387903437_The_Significance_of_Use_Case_Diagrams_in_Software_Development_Keywords_Use_Case_Diagrams_Software_Development_Requirements_Elicitation_System_Design_Stakeholder_Communication_Functional_Requirements_V
[21] ‘ISO 9241-110:2020(en), Ergonomics of human-system interaction — Part 110: Interaction principles’. Accessed: Nov. 21, 2025. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso:9241:-110:ed-2:v1:en
[22] E. Wong, ‘Shneiderman’s Eight Golden Rules Will Help You Design Better Interfaces’, The Interaction Design Foundation. Accessed: Nov. 14, 2025. [Online]. Available: https://www.interaction-design.org/literature/article/shneiderman-s-eight-golden-rules-will-help-you-design-better-interfaces
[23] ‘Web Content Accessibility Guidelines (WCAG) 2.2’. Accessed: Nov. 15, 2025. [Online]. Available: https://www.w3.org/TR/WCAG22/
[24] Asana, ‘What Is Agile Methodology? (A Beginner’s Guide) [2025] • Asana’, Asana. Accessed: Nov. 14, 2025. [Online]. Available: https://asana.com/resources/agile-methodology
[25] ‘What is Scrum? | Scrum.org’. Accessed: Nov. 21, 2025. [Online]. Available: https://www.scrum.org/resources/what-scrum-module

A) Appendix A: System Model and Analysis

Details of Requirements gathering and analysis method

B) Appendix B: Design

Details of design approach used

System elements, components, classes

C) Appendix C: Prompts Used with ChatGPT

D) Appendix D: Additional Code Samples

Provide additional code samples

Organised by Logical Architecture

E) Appendix E:

Other Appendix